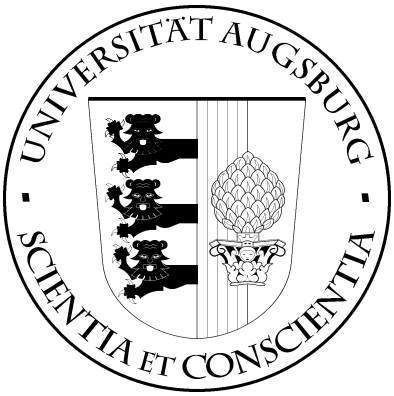
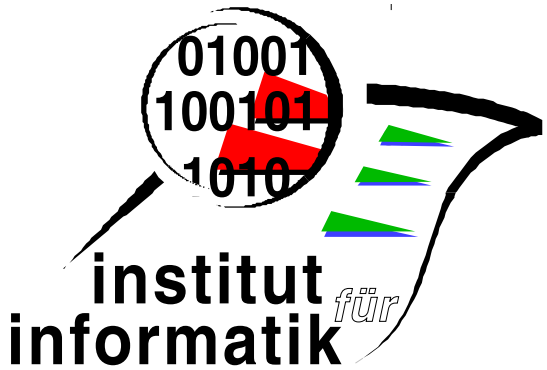
Universität Augsburg



Improved Algorithms for the 2-Vertex Disjoint Paths Problem

Torsten Tholey

Report 2007-16 Dezember 2007



Institut für Informatik
D-86135 Augsburg

Improved Algorithms for the 2-Vertex Disjoint Paths Problem

Torsten Tholey

Institut für Informatik
Universität Augsburg
D-86135 Augsburg, Germany
tholey@informatik.uni-augsburg.de

Abstract. Given distinct vertices s_1, s_2, t_1, and t_2 the 2-vertex-disjoint paths problem consists in determining two vertex-disjoint paths p_1, from s_1 to t_1, and p_2, from s_2 to t_2, if such paths exist.
As a first result we show that by using some kind of sparsification technique the previously best known time bound of $O(n + m\alpha(m, n))$ can be reduced to $O(m + n\alpha(n, n))$, where α denotes the inverse of the Ackermann function. Moreover, we extend the very practical and simple algorithm of Hagerup for solving the 2-vertex-disjoint-paths problem on 3-connected planar graphs to a practical linear time algorithm for the 2-VDPP on general planar graphs thereby avoiding the computation of planar embeddings or triconnected components.

1 Introduction

Given $2k$ pairwise distinct vertices $s_1, \ldots, s_k, t_1, \ldots, t_k$ of a graph $G = (V, E)$, the k-vertex-disjoint paths problem (k-VDPP) consists in determining k pairwise disjoint paths $p_i : s_i \rightarrow t_i$ $(i = 1, \ldots, k)$. This problem is a fundamental routing problem in graph theory which also arises in the context of VLSI-design and network reliability (see [1] and [11]). Unfortunately, for $k \geq 3$, there are still no efficient algorithms for the k-VDPP. More precisely, Fortune, Hopcroft, and Wyllie [4] proved that even the decision version of the 2-VDPP, where we only want to test the existence of disjoint paths without constructing them, is $\mathcal{NP}$-complete on directed graphs. For undirected graphs, the decision version of the k-VDPP is solvable in $O(n^3)$ time for any fixed k as shown by Robertson and Seymour. Perković and Reed [12] improved the running time to the currently best known time bound $O(n^2)$. With a simple reduction increasing the running time to $O(mn^2)$ Perković and Reed's algorithm can be modified not only to test the existence of k disjoint paths, but also to output such paths (cf. [19]). However, the constants hidden in the big-Oh-notation of these algorithms being very large, for $k \geq 3$, there is not yet any known algorithm for the k-VDPP of practical importance. This is why research has been focused on the k-VDPP on restricted sets of graphs (see e.g. [15], [16]), to the case, where $k = 2$ (see below), or both (see e.g. [8], [13]).

Previous results for undirected graphs. Perl and Shiloach [13] found the first linear time algorithm for the 2-VDPP on planar triconnected graphs. Woeginger [22] presented a much simpler algorithm but also used a planar embedding of the graph under consideration. Recently, Hagerup [5] was able to simplify this algorithm to a very practical algorithm on planar triconnected graphs avoiding the computation of planar embeddings. As already stated by Perl and Shiloach in [13] with a reduction of Itai all these algorithms can be modified to solve the 2-VDPP on general planar graphs. However, Itai's reduction makes use of a graph decomposition into triconnected components. Therefore, combining the algorithm of Hagerup with the reduction of Itai does not seem to result in an efficient practical algorithm.

For general undirected graphs, Ohtsuki [11], Seymour [17], Shiloach [18], and Thomassen [20] independently found the first polynomial time algorithms for the 2-VDPP, where Seymour considered the decision version of the problem. The running time of Ohtsuki and Shiloach's algorithm is $O(nm)$. Khuller, Mitchell, and Vazirani [7] by modifying Shiloach's algorithm could prove a running time of $O(n^2)$ for the 2-VDPP. Recently, the author of this paper [19] also following Shiloach's approach reduced the running time to the currently best known time bound $O(n + m\alpha(m, n))$, where α denotes the inverse of the Ackerman function.

New results. In section 2 we sketch the $O(n + m\alpha(m, n))$-time algorithm presented in [19] and then show in section 3 by using some kind of sparsification technique how it can be modified to run in $O(m + n\alpha(n, n))$ time. In section 4 we extend the algorithm of Hagerup [5] for triconnected planar graphs to general planar graphs resulting in the first practical algorithm for the 2-VDPP on planar graphs which avoids the computation of both planar embeddings and triconnected components.

Further in this section we introduce some notations used in this paper. For a graph G, we define E_G as the edge set of G and V_G as the vertex set of G. We write $\deg_G(v)$ for the degree of v in G. Throughout this paper, paths will always mean simple paths, i.e. paths visiting each of its vertices at most once. Therefore, for a path p and two vertices a and b of p, we can denote by $p[a, b]$ the subpath of p, or of the reverse path of p, leading from a to b. For a vertex v, an edge e, and a path p, we write $v \in p$ or $e \in p$ if v or e, respectively, is part of p. By $p \circ q$ we denote the concatenation of two paths p and q.

For a graph G, we call a set $S \subseteq V_G$ a *separator of* G if $G - S$ is not connected, and a separator S a *k-separator* if $|S| = k$. We say that two vertices v and w are separated by $S \subseteq V_G$ if they lie in different connected components of $G - S$. Two vertices v and w are *k-connected* if there are k *internally disjoint paths* between v and w, i.e. k paths such that no pair of paths has a vertex outside $\{v, w\}$ in common. From Menger's theorem it is well-known that non-adjacent vertices v and w are k-connected iff there is no $(k-1)$-separator separating v and w.

In the following we denote an instance of the 2-VDPP by a tuple $I = (G, s_1, s_2, t_1, t_2)$ with G being a graph and s_1, s_2, t_1 and t_2 being the vertices of G for which we search for two disjoint paths p_1, from s_1 to t_1, and p_2, from s_2 to t_2.

2 Solving the 2-VDPP in $O(n + m\alpha(m, n))$ Time

In the following we give a sketch of the $O(n + m\alpha(m, n))$-time algorithm for the 2-VDPP presented in [19].

For an instance $I = (G, s_1, s_2, t_1, t_2)$ of the 2-VDPP, let $G_{x,I}$ be the graph obtained from G by adding an extra vertex x as well as edges $(x, s_1), (x, s_2), (x, t_1)$, and (x, t_2). The following property P may or may not hold for I:

(P) All vertices in G are 4-connected to x in $G_{x,I}$.

Shiloach [18] showed that the 2-VDPP on a triconnected graph G with property P can be solved in linear time plus possibly (in the case of G being not planar) the time needed to find a subgraph of G homeomorphic to K_5 or $K_{3,3}$. Williamson [21] presented a linear time algorithm for the latter problem. Since Itai found a linear time reduction from the 2-VDPP on general graphs to the 2-VDPP on triconnected graphs sketched in [18], for obtaining an efficient algorithm for the 2-VDPP we only have to search for an efficient algorithm reducing an instance of the 2-VDPP on a triconnected graph to an instance on a triconnected graph with property P. For such a reduction, we repeatedly search for vertices not being 4-connected to x and remove them by using so-called Δ-replacements:

Let S be a 3-separator separating v and x in $G_{x,I}$. By a *Δ-replacement of G by S deleting v* we mean the replacement of G by the graph G' obtained by deleting all vertices of the connected component of $G - S$ containing v with their adjacent edges and inserting edges between the vertices of S that are not already connected by an edge. Then the following Lemma holds (see [18] and [19]).

Lemma 1. *Let $I = (G, s_1, s_2, t_1, t_2)$ be an instance of the 2-VDPP on a triconnected graph G and S be a 3-separator separating a vertex v and x in $G_{x,I}$. Then the graph G' resulting from a Δ-replacement of G by S deleting v is triconnected and contains s_1, s_2, t_1, t_2. Moreover, the 2-VDPP on I has a solution iff the same is true of $I' = (G', s_1, s_2, t_1, t_2)$. Finally the degree of connectivity between x and any vertex w not being deleted by the Δ-replacement can be increased but not decreased by the replacement.*

For determining vertices not 4-connected to x the $O(n + m\alpha(m, n))$-time algorithm in [19] uses a data structure D of Kanevsky et. al. [6]. This data structure being initialized with a triconnected graph supports the insertion of additional edges, can answer queries whether two vertices v and w in the current graph are 4-connected, and finally, if they are neither adjacent nor 4-connected it can output a 3-separator separating v and w. One can use this data structure to identify a vertex v in $G_{x,I}$ not 4-connected to x (and hence non-adjacent to x since one can show that the vertices s_1, s_2, t_1, and t_2 adjacent to x are 4-connected to x) and a 3-separator S separating v and x, and then run a Δ-replacement by S deleting v. However, since D does not support edge deletions, we use data structure D instead for $G_{x,I}$ for a graph H that only before the first Δ-replacement should be equal to $G_{x,I}$. After each Δ-replacement the same

edges are added to $G_{x,I}$ and H, but no edge of H is deleted. We can do so since at any step of our algorithm a vertex v is 4-connected to x in $G_{x,I}$ iff it is 4-connected to x in H and each 3-separator S separating v from x in H also separates v and x in $G_{x,I}$ (see [19]). A high-level implementation of the reduction described above is shown in Fig. 1.

```
(1) For each v ∈ V
(2)     Using D test whether v is 4-vertex-connected to x in H.
(3)     If v is not 4-vertex-connected to x in H:
(4)         Using D find a 3-separator S seperating v from x in H.
(5)         Modify G and G_{x,I} by a Δ-replacement by S deleting v.
(6)         For each edge e inserted into G by the Δ-replacement
(7)             Insert e into H.
(8)             Update the dynamic data structure D.
(9) Return G
```

Fig. 1. Reducing an instance of the 2-VDPP to an instance with property (P).

By Lemma 1 the algorithm shown in Fig. 1 determines a graph G^* with property P that contains disjoint paths $p_1 : s_1 \to t_1$ and $p_2 : s_2 \to t_2$ iff the same is true for the original graph G. Given such paths in G^* it is possible, in some kind of backtracking steps, to compute two paths solving the original instance in $O(n + m)$ time (for details see [19]). Concerning the running time of the reduction to G^*, D supports the initialization with a triconnected graph in $O(n + m\alpha(m, n))$ time and up to $O(n)$ queries, computations of 3-separators, and edges insertions in $O(n\alpha(m, n))$ time. Since one can show that, given v and S, the running time of a Δ-replacement can be bounded to be at most linear in the number of edges deleted by the replacement, the whole reduction runs in $O(n + m\alpha(m, n))$ time (compare [19]).

The main idea of our new algorithm is to replace the graph H by a so called *sparse certificate*. For a graph G and a graph property $\mathcal{P}$, a graph G' is a *certificate* of $(G, \mathcal{P})$ if G' has property $\mathcal{P}$ if and only if G has property $\mathcal{P}$. A certificate is a *sparse certificate* if it has $O(n)$ edges. Eppstein, Galil, Italiano, and Nissenzweig in [2] showed that for many algorithms with a running time of the form $O(f(n, m))$ dynamically updating a graph by edge insertions and deletions, sparse certificates can help to reduce the running time to $O(f(n, n))$.

We will use two kinds of special sparse certificates. A *sparse certificate for the k-connectivity* of a graph G is a subgraph G' of G with $V_{G'} = V_G$ and $O(n)$ edges, where for each pair of vertices v and w of G, v and w are k-connected in G if and only if they are k-connected in G'. A *sparse certificate for the k-connectivity to a vertex x* of a graph G containing x is a subgraph G' of G with $V_{G'} = V_G$ and $O(n)$ edges, where a vertex v is k-connected to x in G if and only if the same is true of G'.

Unfortunately, updating a sparse certificate for the k-connectivity to a vertex x after a Δ-replacement is not easy. In order to solve this problem we choose the vertices not being 4-connected to x in line 2 of the algorithm in Fig. 1 in a particular order and we sometimes modify a vertex v and/or a 3-separator computed by the algorithm of Fig. 1 in order to guarantee that the graph H after the update is a sparse certificate for the 4-connectivity to x for the graph $G_{x,I}$ after the update.

3 An $O(m + n\alpha(n, n))$-time Algorithm for the 2-VDPP

We present an $O(m+n\alpha(n,n))$-time reduction of an instance $I = (G, s_1, s_2, t_1, t_2)$ of the 2-VDPP on a triconnected graph G to an instance $I' = (G', s_1, s_2, t_1, t_2)$ with G' being a triconnected graph with property P (defined in the previous section). The reduction is divided in two phases. We start with the description of the first phase.

The first phase dynamically updates $G_{x,I}$ as well as two graphs H and H_* with H being initialized with a copy of $G_{x,I}$, and H_* being initialized with a sparse ceritificate for the k-connectivity of H for all $k \leq 4$. H is not really needed and used only to simplify the proof of correctness. With the algorithm of Nagamochi and Ibaraki [10] H_* can be initialized in linear time. During each update, all edges inserted into $G_{x,I}$ are also inserted into H and H_*, but no edge will be deleted from H and H_*. We also construct the data structure of Kannevsky et al [6] described in the previous section in order to test the four-connectivity between two vertices in H_*, and we call it D_*. Finally, we maintain a list L of all $v \in V_G$ with $\deg_G(v) > 3$. Since a Δ-replacement only changes the degree of a constant number of vertices remaining in G, – beside restricting L to the remaining vertices – L can be updated in $O(1)$ time after the Δ-replacement.

After initializing H, H_*, D_*, and L we use D_* to test, for each vertex $v \in L$, whether it is 4-connected to x. If not, again using D_*, we determine a 3-seperator S separating x and v in H_*. Similar to the algorithm of Fig. 1, we want to run a Δ-replacement by a 3-separator S', but we choose S' very carefully instead of just setting $S' = S$. Let K be the subgraph of H_* induced by the vertices of S and the vertices of the connected component of $H_* - S$ containing v. Let S' be the set obtained from S by replacing each vertex $w \in S$ with $\deg_K(w) = 1$ by its unique neighbor $n(w)$ in K. A unique neighbor $n(w)$ of a vertex $w \in S$ with $\deg_K(w) = 1$ cannot be equal to another vertex $u \in S$ or, if $\deg_K(u) = 1$, to its unique neighbor $n(u)$ in K (otherwise $S - \{w\}$ or $(S - \{u, w\}) \cup \{n(w)\}$ would be a 2-separator in the triconnected graph H_*). Therefore, we have $|S'| = 3$. We will later show that a 3-separator separating a vertex w from x in H_* also separates x and w in $G_{x,I}$. From $K - S$ containing v and $\deg_G(v) > 3$ we then can conclude that with S separating v and x in G set S' separates a vertex v' equal to v or a neighbor of v from x in $G_{x,I}$. Moreover, for the subgraph K' of H_* induced by the vertices of S' and the vertices of the connected component of $H_* - S'$ containing v', $\deg_{K'}(w) \geq 2$ holds for all $w \in S'$. We now run a Δ-replacement on $G_{x,I}$ by S' deleting v'. If v remains in H_* and $G_{x,I}$, we have $v \in S' - S$ and

```
(1) Construct a list L of all vertices v with deg_G(v) ≥ 4
(2) For each v ∈ L
(3)     Using D_* test whether v is 4-vertex-connected to x in H_*.
(4)     If v is not 4-vertex-connected to x in H_*:
(5)         Using D_* find a 3-separator S separating v and x in H_*.
(6)         K := subgraph of H_* induced by S and the vertices of
                 the connected component of H_* − S containing v.
(7)         For each v ∈ V_S
(8)             If v has only one neighbor u in K
(9)                 S := (S − {v}) ∪ {u}   /* replacement of S by S'. */
(10)        Let v' be equal to v or a neighbor of v separated by S from x in G_{x,I}
(11)        Modify G and G_{x,I} by a Δ-replacement by S deleting v'.
(12)        For each edge e inserted into G by the Δ-replacement
(13)            Insert e into H and H_*.
(14)            Update list L and the dynamic data structure D_*.
(15) Return G
```

Fig. 2. A high-level implementation of phase 1.

v is a neighbor of exactly one vertex $w \in S$. Hence, after the replacement v has exactly three neighbors, namely w and the vertices in $S' - \{w\}$. Since we have $\deg_{K'}(w) \geq 2$ for all $w \in S'$ the degree of a vertex remaining in $G_{x,I}$ cannot increase after the Δ-replacement by S' and, similiarly, also after all following Δ-replacements. This also being true of v vertex v is never reinserted to L in phase 1. Fig. 2 shows a high-level implemenation of phase 1.

At the end of phase 1 $G_{x,I}$ should be triconnected and (G, s_1, t_1, s_2, t_2) should be solvable iff our original instance is solvable. Finally, a vertex v in G should be 4-connected to x in G iff $\deg(v)_{G_{x,I}} \neq 3$. All this follows from the considerations above and Lemma 1, if the first two of the following properties hold:

(1) v is 4-connected to x in $G_{x,I}$ iff the same is true for H_*,
(2) If $v \in V_G$ is not 4-vertex-connected to x in H_*, a 3-separator S separating v and x in H_* is also a 3-separator separating v and x in H and $G_{x,I}$.
(3) Two vertices v and w are 4-vertex connected in H iff the same is true of H_*.
(4) $G_{x,I}, H$, and H_* are triconnected.

However, for proving property (2) we need the properties (3) and (4):

Lemma 2. *Before or after a Δ-replacement, property (2) follows from the properties (3) and (4).*

Proof. As shown in [19], if, for each Δ-replacement of G, H is updated by inserting the same edges as into G but not deleting any edge, a 3-separator S separating a vertex $v \in V_{G_{x,I}}$ and x in H also separates v and x in $G_{x,I}$. It remains to show that a 3-separator S separating a vertex $v \in V_{G_{x,I}}$ from x in H_* also separates v and x in H.

Suppose there is a 3-separator S separating a vertex $v \in V_G$ from x in H_* but not in H. Then there must be an edge $(u, w) \in E_H$ such that u and w lie in different components of $H_* - S$ and $(u, w) \notin E_{H_*}$.

Since H_* is triconnected (property (4)) there are three disjoint paths from u to w in H using only edges of H_*. Edge (u, w) defines a fourth path from u to w so that u and w are 4-connected in H but not in H_*, a contradiction to (3).

Lemma 3. *Properties (1), (3), (4) hold before and after each Δ-replacement.*

Proof. Before the first Δ-replacement properties (1), (3), and (4) hold since at this time H is a copy of $G_{x,I}$ and H_* is a sparse certificate of the k-connectivity for H for every $k \leq 4$.

Let us assume that properties (1), (3), and (4) and hence also property (2) (by Lemma 2) hold before a Δ-replacement of G by a 3-separator S' deleting a vertex v'. Let $G_{x,I}, H$, and H_* be the graphs before the Δ-replacement and $G'_{x,I}, H', H'_*$ be the corresponding graphs after the Δ-replacement.

Since no edge of H and H_* is deleted, H' and H'_* are triconnected. By Lemma 1 $G'_{x,I}$ is also triconnected. In [19] the author of this paper has shown that a vertex $v \in V_{G'_{x,I}}$ is 4-connected to x in $G'_{x,I}$ iff the same is true of H'. Hence property (1) follows from property (3). It remains to show property (3).

Since $E_{H'_*} \subseteq E_{H'}$, two vertices 4-connected in H'_* are also 4-connected in $E_{H'}$. For the reverse direction, let p_1, p_2, p_3, and p_4 be four internally disjoint paths in H' leading from a vertex u to a vertex w. It remains to show that there are also four disjoint paths from u to w in H'_*. Before showing that, let us define $K_{v'}$ to be the connected component of $G_{x,I} - S'$ deleted from $G_{x,I}$ by the Δ-replacement, i.e. containing v'. Moreover, we let K_x be the union of all other components remaining in G. For each pair of vertices $y, z \in S'$ and each $C \in \{K_{v'}, K_x\}$, let $p(y, z, C)$ be a path from y to z in H_* that, except y and z, only visits vertices of C. Such paths exist: Note that there are three paths from v' to the vertices of S' in $K_{v'}$ that with the exception of all paths visiting v' are disjoint, since S' is a 3-separator in the triconnected graph H_*. Hence, we can construct $p(x, y, K_{v'})$ by combining the two of the three paths ending in x and y. The existence of a path $p(x, y, K_x)$ can be shown in a similar way. We now distinguish several cases depending on the cardinality of $S' \cap \{u, w\}$.

If $\{u, w\} \subseteq S'$, then there are four disjoint paths from u to w in H'_*, namely, if y is the vertex of $S' \setminus \{u, w\}$, the paths $(u, w), (u, y) \circ (y, w), p(x, y, K_{v'})$, and $p(x, y, K_x)$.

If $\{u, w\} \not\subseteq S'$ and at most one path p out of p_1, p_2, p_3, and p_4 visits an edge with an endpoint in $K_{v'}$ or a newly inserted edge between two vertices of S', let a be the first and b be last vertex of S' visited by p. By replacing $p[a, b]$ by $p(a, b, K_{v'})$ we obtain four internally disjoint paths from u to w in H instead of H'. Since (3) holds before the Δ-replacement there are also 4 internally disjoint paths from u to w in H_* and hence also in H'_* ($E_{H_*} \subseteq E_{H'_*}$).

We now examine the case $\{u, w\} \not\subseteq S'$ with two paths p, q out of the four disjoint paths p_1, p_2, p_3, p_4 from u to w visiting edges with an endpoint in $K_{v'}$ or a newly inserted edge between the vertices of S. This can only be the case if

$|\{u, w\} \cap S'| = 1$, say w.l.o.g. $w \in S'$. Let a and b be the first vertices of S' visited by p or q, respectively. Suppose that we can show that the subpaths $p[a, w]$ and $q[b, w]$ can be replaced by paths p', from a to w, and q', from b to w, in $H - K_x$ that except both paths visiting w are disjoint. Then, similiar to the previously considered case, we obtain four internally disjoint paths from u to w in H and hence also in H_* and H'_*.

Guaranteeing the existence of p' and q' is the reason for dividing our algorithm into two phases. We have already shown that, by our choice of v' and S', for the subgraph K of H induced by the vertices of $K_{v'}$ and S', $\deg_K(x) \geq 2$ holds for all $x \in K$ (more precisely, we considered the subgraph H_* of H instead of H). If (a, w) or (b, w) is an edge in H, say w.l.o.g. (a, w), then, if (b, w) is an edge in H, we can define $p' := (a, w)$ and $q' := (b, w)$, and otherwise b is connected to a vertex c of $K - S'$ different from a and w. In the latter case, because of the triconnectivity of H there exists a path $\tilde{p}$ from c to w not visiting a or b and which is completely contained in K. In this case we can define $p' := (a, w)$ and $q' := (b, c) \circ \tilde{p}[c, w]$.

If neither the edge (w, a) nor the edge (w, b) exists in H, w is connected to two distinct vertices c, d in $K - S'$. Then there are three disjoint paths $p_{a,c}, p_{b,c}, p_{w,c}$ leading from a, b, or w, respectively, to c as well as three disjoint paths $p_{a,d}, p_{b,d}, p_{w,d}$ leading from a, b, or w to d in K. Choose $\tilde{p}$ as one of the two paths $p_{a,d}$ and $p_{b,d}$ such that $\tilde{p}$ does not visit c. Say w.l.o.g. $\tilde{p} = p_{a,d}$. If $\tilde{p}$ is internally disjoint to $p_{a,c}$ and disjoint to $p_{b,c}$, we define $p' := \tilde{p} \circ (d, w)$ and $q' := p_{b,c} \circ (c, w)$. Otherwise, let be y the last vertex of $p_{a,d}$ which is also visited by $p_{a,c}$ and $p_{b,c}$, say w.l.o.g. by $p_{a,c}$. Then we define $p' := p_{a,c}[a, y] \circ \tilde{p}[y, d] \circ (d, w)$ and $q' := p_{b,c} \circ (c, w)$. This completes our proof.

At the end of phase 1 a vertex v in $G_{x,I}$ is 4-connected to x iff $\deg_{G_{x,I}}(v) \geq 4$. We now start phase 2 as phase 1 with constructing a copy H of $G_{x,I}$ and a sparse certificate H_* for the k-connectivity of H for all $k \leq 4$. Immediately after the initialization of H and H_* we color all vertices $v \in V_G$ with $\deg_{G_{x,I}}(v) \geq 4$ black and the remaining vertices w with $\deg_{G_{x,I}}(w) = 3$ red. The color of a vertex will never be updated in phase 2. According to the triconnectivity of G the vertices s_1, s_2, t_1, and t_2 are 4-connected to x in $G_{x,I}$ and hence are colored black.

In order to delete the remaining vertices of $G_{x,I}$ not 3-connected to x, we use the algorithm of Fig. 1, but we replace each occurrence of H by H_* (more precisely, for the sake of analysis, let us assume that in line 7 we insert e also into H and not only into H_*). This works if the following invariant (2) holds. We prove this by showing that the following invariants (1) and (2) hold before and after each Δ-replacement.

(1) Each edge of $G_{x,I}$ adjacent to a red colored vertex is also contained in H_*.
(2) Two vertices $v, w \in V_{G_{x,I}}$ are 4-vertex-connected in H_* if and only if the same is true of $G_{x,I}$. Moreover, every 3-separator separating a vertex $v \in V_{G_{x,I}}$ from x in H_* is also a 3-separator separating v and x in H and $G_{x,I}$.

Lemma 4. *Invariant (1) holds before and after each Δ-replacement.*

Proof. H_* being initialized with a sparse certificate for the 3-connectivity of the original graph $G_{x,I}$, the current graph H_* contains all edges of the original graph $G_{x,I}$ adjacent to a red colored vertex. All other edges adjacent to a red colored vertex in the current graph $G_{x,I}$ have been inserted by a Δ-replacement. By construction, they are also inserted into H_*.

We will prove invariant (2) by using the following lemma:

Lemma 5. *Before or after a Δ-replacement, if (2) holds, each vertex of a 3-separator S separating a vertex v and x in $G_{x,I}$ is either a red colored vertex or a neighbor of a red colored vertex.*

Proof. Assume that after k Δ-replacements ($k \in \mathbb{N}_0$) there is 3-separator S separating a vertex v and x in $G_{x,I}$ and that S contains a black colored vertex w not adjacent to a red vertex. By (2) v is not 4-connected to x in the current graph H_*. Since the degree of connectivity between two vertices in H_* cannot decrease, v is a red vertex. Since $G_{x,I}$ is triconnected, there are internally disjoint paths p_1, p_2, and p_3 from v to x. Let b_1, b_2, b_3 the first black colored vertices on these paths. b_1, b_2, and b_3 exist, since s_1, s_2, t_1, t_2 are colored black. One path among $p_1[v, b_1], p_2[v, b_2]$, and $p_2[v, b_3]$ must be completely contained in $G_{x,I} - S$, since by definition none of these paths can visit w and $|S - \{w\}| = 2$. Say w.l.o.g. $p_1[v, b_1]$ is contained in $G_{x,I} - S$. As a black vertex there are four internally disjoint paths between x and b_1 in $G_{x,I}$ and H_* before the first Δ-replacement. Consequently, this also holds for the graph H_* after k Δ-replacements and by (2) also for the graph $G_{x,I}$. Hence, there are four internally disjoint paths from b_1 to x with one of them not visiting any vertex in S. The concatenation of this particular path and $p_1[v, b_1]$ results in a path from v to x in $G_{x,I} - S$; a contradiction to the fact that S separates v and x in $G_{x,I}$.

Corollary 6. *Invariant (2) holds before and after each Δ-replacement.*

Proof. As shown in [19], a vertex is 4-connected to x in $G_{x,I}$ iff the same is true for H. Therefore, a vertex v 4-connected to x in H_* is also 4-connected to x in H and $G_{x,I}$. Assume now that there is a 3-separator S separating a vertex v and x in H_* but not in $G_{x,I}$ (or not in H). Since during the Δ-replacements the degree of connectivity between two vertices cannot decrease in H_*, v is a red vertex. Let p be a path from v to x in $G_{x,I} - S$ (or in $H - S$) and let b be the first black vertex on this path. According to invariant (1) $p[v, b]$ is completely contained in H_*. For b as a black vertex there must exist four internally disjoint paths from b to x in H_* with one of them not visiting any vertex of S. Hence, there are four paths from v to b and from b to x in $H_* \setminus S$, a contradiction to the fact that S separates v and x in H_*.

At the end of phase 2 we obtain a graph $G_{x,I}$ with property P (this follows from the correctness of the algorithm of Fig. 1 and from invariant (2)).

The analysis of the running time of our reduction algorithm is very similar to the one already used by the author in [19]. The time needed for the initialization of the dynamic data structure D_* with H_* as well as the time needed to answer

a total of up to n queries whether two vertices are 4-connected and to compute up to n 3-separators, can be bounded by $O(n\alpha(n,n))$ time, since H_* has only $O(n)$ edges. Since the running time of a Δ-replacement, takes time linear in the number of edges deleted by the Δ-replacement and since the remaining parts of our algorithm run in linear time, the whole reduction algorithm runs in $O(m + n\alpha(n,n))$ time.

4 Solving the 2-VDPP on Planar Graphs

In this section we want to show how the very practical algorithm of Hagerup [5] for the 2-VDPP on triconnected planar graphs can be extended to a practical algorithm for the 2-VDPP on planar graphs. Starting on an instance $I = (G, s_1, s_2, t_1, t_2)$ the algorithm of Hagerup uses the triconnectivity of G only to guarantee the existence of three internally disjoint paths from s_1 to t_1 and three internally disjoint paths from s_2 to t_2. Hence, we only need to show how an instance $I = (G, s_1, s_2, t_1, t_2)$ of the 2-VDPP on a general planar graph can be reduced to an instance $I' = (G', s'_1, s'_2, t'_1, t'_2)$ with three internally disjoint paths between s'_1 and t'_1 as well as between s'_2 and t'_2.

Say w.l.o.g. that G is connected. If there are less than three disjoint paths between s_1 and t_1, we will split our graph into smaller subgraphs: For a graph H and a separator S separating vertices v and w in H, let $H_S(v)$ be the graph obtained from the subgraph of H induced by the vertices of S and the vertices of the connected component C of $H - S$ containing v by inserting edges between the vertices of S that are not already connected. Define $H_S(\neg v)$ to be the graph obtained from the subgraph of H induced by the vertices of $V_H \setminus V_C$ by inserting edges between all vertices of S that are not already connected. For splitting an instance into smaller instances we will apply the following lemma:

Lemma 7. *Let $k \geq 1$ be an arbitrary but fixed constant. Given two non-adjacent vertices s and t and k disjoint paths $p_1, \ldots, p_k$ from s to t in a graph G, we can find either $(k+1)$ disjoint paths from s to t in $O(m)$ time or a k-separator S of G separating s and t with no vertex $x \in S$ being separated by a k-separator from s in $G_S(s)$ in $O(|E_{G_S(s)}|)$ time.*

Proof. We number the vertices of each path p_i with increasing numbers in their order of appearance on p_i and we define for each path p_i a so-called *stop vertex* initialized with the vertex adjacent to s on p_i. Each stop vertex u is a possible candidate for being part of a k-separator separating s and t and is always chosen as a vertex u lying on p_i from which we know that no vertex before u on p_i can be contained in a k-separator separating s and w. It is clear that this is true immediately after the initialization of the stop-vertices. We then start a depth-first search not visiting the stop vertices until this search reaches a vertex w of a path p_i with $i \in \{1, \ldots, k\}$ and with a number $n(w)$ higher than the number $n(u)$ of the current stop vertex u of p_i. In this case we can conclude that the current stop vertex u and all vertices x appearing between u and w on p (except w itself) cannot be part of a k-separator S: Otherwise, S cannot

contain another vertex of p_i since S clearly contains exactly one vertex of each of the paths $p_1, \ldots, p_k$. Moreover, our depth-first search guarantees that there is a path p from s to w which, apart from w, on $p_1, \ldots, p_k$ only visits vertices from which we already know by induction that they cannot be part of a k-separator. Therefore, $p \circ p_i[w, t]$ would be a path from s to t in $G \setminus S$ contradicting the fact that S is a separator. Hence, we redefine the stop vertex of p_i to be the vertex w, continue the depth-first search with the vertex visited immediately before w now not visiting the new stop vertices and add u to a list L of old stop vertices.

If this depth-first search stops, it has possibly explored neither G nor the complete graph $G \setminus S$ with S being the set of current stop vertices, since some of the old stop vertices may not have been visited. However, with the help of list L it is easy to continue the depth-first search on these vertices. If we also continue to dynamically update the list L we either visit the whole graph G or the whole graph $G \setminus S$ with S being the set of current stop vertices.

If the depth-first search reaches t, we can conclude, as above, that there is no k-separator separating s and t and we can construct $k+1$ disjoint paths between s and t with the classical algorithm of Ford und Fulkerson [3].

Otherwise, the set S of the current stop vertices defines a k-separator separating s and t. Moreover, there is no k-separator S' of $G_S(s)$ separating s from a vertex in S: Otherwise, because of the edges between the vertices of S it must contain, for each $i \in \{1, \ldots, k\}$, a vertex s_i on p_i. But then it is easy to see that S' also separates s and t in G. Moreover, the vertices of S' do not appear after the vertices of S on $p_1, \ldots, p_k$ and S' is not equal to S. However, by our definition of stop vertices such a k-separator separating s and t cannot exist.

Concerning the running time, for finding a k-separator S the depths-first search only visits vertices in $G_S(s)$ and therefore runs in either $O(|E_{G_S(s)}|)$ or $O(|E_G|)$ time depending on whether or not a k-separator is found.

By splitting instances of the 2-VDPP according to Lemma 7 we obtain the following lemmas.

Lemma 8. *Suppose that the 2-VDPP can be solved in linear time on instances (G, s_1, s_2, t_1, t_2), where G is a planar graph with two disjoints paths between s_1 and t_1 as well as between s_2 and t_2. Then the 2-VDPP on general planar graphs can also be solved in linear time.*

Proof. We first show how to guarantee the existence of two disjoint paths between s_1 and t_1. Let (G, s_1, s_2, t_1, t_2) be an instance of the 2-VDPP on an arbitrary planar graph. W.l.o.g. let us assume that G is connected. Moreover, I should be *non-simple*, i.e $s_1 \neq t_1$ and $s_2 \neq t_2$. Otherwise, I can be solved easily by a depth-first search on $G \setminus \{s_1\}$ or $G \setminus \{s_2\}$, respectively. We first construct a simple path from s_1 to t_1. According to Lemma 7 we can either find two disjoint paths from s_1 to t_1 in $O(|E_G|)$ time, or we can determine a vertex v with $S := \{v\}$ separating s_1 and t_1 and with no 1-separator separating v and s_1 in $O(|E_{G_S(s_1)}|)$ time. Hence, assume the latter is the case.

If the connected component C of $G - \{v\}$ containing s_1 contains exactly one of the vertices s_2 and t_2, obviously, there are no two disjoint paths solving our instance of the 2-VDPP. Otherwise, if C contains both, s_2 and t_2, we can reduce our original instance to $I := (G_S(v), s_1, s_2, v, t_2)$. More precisely, for two disjoint paths p_1, from s_1 to v, and p_2, from s_2 to t_2, on $G_S(v)$, we can extend p_1 by a path from v to t_1 in $G_S(\neg v)$ in $O(|E_{G_S(\neg v)}|)$ time.

Similarly, if C contains neither s_2 nor t_2 our problem can be reduced to $I' = (G_S(\neg v), v, s_2, t_1, t_2)$ which can be solved recursively.

After guaranteeing the existence of two disjoint paths between one pair of a source and a target vertex we can apply the same reduction to guarantee the existence of two disjoint paths between both pairs of source and target vertices.

In our next step we show:

Lemma 9. *Suppose that the 2-VDPP can be solved in linear time on instances (G, s_1, s_2, t_1, t_2), where G is a planar graph with three disjoint paths between s_1 and t_1 as well as two disjoint paths between s_2 and t_2. Then the 2-VDPP on general planar graphs can also be solved in linear time.*

Proof. According to Lemma 8 we only want to consider a non-simple instance $I := (G, s_1, s_2, t_1, t_2)$ on a planar graph G with two disjoint paths between s_1 and t_1 as well as two disjoint paths between s_2 and t_2. With the classical algorithm of Ford and Fulkerson we can construct two disjoint paths p_1 and p_2 or, if they exist, three disjoint paths p_1, p_2, and p_3 from s_1 to t_1 in linear time. If three disjoint paths do not exist, we can split the original instance into smaller instances with three disjoint paths between one pair of a source and target vertex as follows:

Starting with the original instance I during our algorithm we will always encounter either exactly one instance $I_1 = (H, s_1', s_2', t_1, t_2')$ or two instances $I_1 = (H, s_1', s_2', t_1, t_2')$ and $I_2 = (H, s_1'', s_2', t_1, t_2')$. Except immediately after the initialization, where $H = G$ and $s_1' = s_1$, we have $H = G_T(\neg s_1)$, where T is a 2-separator computed in a previous round separating s_1 and t_1. We will also have $s_1' \in T$, and if s_1'' exists, $T = \{s_1', s_1''\}$. Applying Lemma 7 we will search for a 2-separator separating s_1' and t_1 and hence also separating s_1 and t_1 in order to split instance I_1 as well as I_2, if the later exists, i.e. we try to find solutions for I_1 and I_2 simultaneously. Hence, let us assume that we are faced with one or two instances I_1 or I_1 and I_2 as described above and (except in Case 4) we found a set $S = \{u, w\}$ separating s_1' and t_1 with $u \in p_i$ and $w \in p_j$ and $\{i, j\} = \{1, 2\}$.

Case 1 $H_S(\neg s_1') = G_S(\neg s_1)$ contains both, s_2' and t_2'. In this case we will always have $s_2' = s_2$ and $t_2' = t_2$. We reduce our problem to the instances $I_3 = (G_S(\neg s_1'), u, s_2', t_1, t_2')$ and $I_4 = (G_S(\neg s_1'), w, s_2', t_1, t_2')$. Given two paths p and q solving the 2-VDPP on one of these instances with p leading from u or w, respectively, to t_1, p can by extended by $p_i[s_1', u]$ or $p_j[s_1', w]$, respectively, to a solution of I_1. More precisely, since we may have $(u, w) \notin E_G$, p should not visit (u, w). However, if p visits both, u and w, p can be shortend by removing $p[u, w]$ or $p[w, u]$, respectively, since after the removal p and q still define a solution to one of the instances I_3 and I_4. Because of edge (s_1', s_1'') we can also find two

paths solving I_2 with none of these paths visiting both, u and w. Conversely, if neither I_3 nor I_4 has a solution, neither I_1 nor I_2 can have a solution.

Case 2 $H_S(\neg s'_1)$ contains exactly one of s'_2 and t'_2, say w.l.o.g. t'_2. Then, for two paths p and q solving the 2-VDPP on I_1 (or on I_2), it is obvious that the parts of the paths p and q contained in $H_S(s'_1)$ must solve the 2-VDPP on one of the instances $J_1 = (H_S(s'_1), s'_1, s'_2, u, w)$ and $J_2 = (H_S(s'_1), s'_1, s'_2, w, u)$ (or one of the instances $J_3 = (H_S(s'_1), s''_1, s'_2, u, w)$ and $J_4 = (H_S(s'_1), s''_1, s'_2, w, u)$). Let $(H_S(s'_1), \tilde{s}_1, \tilde{s}_2, \tilde{t}_1, \tilde{t}_2)$ be the instance J_x for an $x \in \{1, \ldots, 4\}$. Solving J_x is easier than solving I_1 or I_2: Either J_x is simple or, since by construction there is no 2-separator separating s'_1 from $\tilde{t}_1$–i.e. three disjoint paths between s'_1 and $\tilde{t}_1$–J_1 and J_2 can be solved in linear time. For the instances J_3 and J_4, we know that there is no 2-separator separating $\tilde{s}_1 = s''_1$ and $\tilde{t}_1$ not containing s'_1. Unfortunately, since there may be 2-separators containing s'_1 and separating s''_1 from $\tilde{t}_1$ the instances J_3 and J_4 have to be solved recursively by our reduction algorithm. However, given J_x and a 2-separator $S' := \{s'_1, w'\}$, when splitting J_x into instances on the graphs $H_{S'}(s''_1)$ and $H_{S'}(\neg s''_1)$ we will show that we only have to search for a solution of an instance of the 2-VDPP on $H_{S'}(s''_1)$ with the path starting in s''_1 leading to w' and not to s'_1 and hence on $H_{S'}(\neg s''_1)$ for a solution with the path ending in $\tilde{t}_1$ starting in w' and not in s'_1. This means that for solving J_x we need not to solve two but only one instance on $H_{S'}(s''_1)$ as well as on $H_{S'}(\neg s''_1)$. Note that the instance on $H_{S'}(s''_1)$ can be solved in linear time since there is no 2-separator separating s''_1 and w'. The instance on $H_{S'}(\neg s''_1)$ can be recursively handeld in the same way as instance J_x. Hence, our reduction algorithm splits J_x only in a linear number of instances of the 2-VDPP on subgraphs that except the constructed 2-separators are distinct. Therefore, a solution for J_3 as well as for J_4 can be found in linear time. It remains to show that on the subgraph $H_{S'}(s''_1)$ we do need to consider solutions of the 2-VDPP with s''_1 leading to s'_1. This is obvious if s'_1 is equal to either s'_2 or t'_2. Otherwise because of s'_2 and t'_2 lying in $H - \{s'_1, s''_1\}$ and $T := \{s'_1, s''_1\}$ being a 2-separator determined in a previous step, in the step finding the 2-separator T, we must have been in Case 1 and must have reduced the problem to the problem of solving one of our instances $I_1 = (G_T(\neg s_1), s'_1, s'_2, t_1, t'_2)$ and $I_2 = (G_T(\neg s_1), s''_1, s'_2, t_1, t'_2)$. But for Case 1 we already know that with a solution to one of these instances there also exist two paths solving one of these instances with none of these paths visiting both, s'_1 and s''_1.

If, for $x \in \{1, 3\}$, there are two paths p'_1 and q'_1 solving J_x but no two paths p'_2 and q'_2 solving J_{x+1}, it is easy to see that $I_{\lceil \frac{x}{2} \rceil}$ can be reduced to the instance $J'_1 := (H_S(\neg s'_1), u, w, t_1, t'_2)$. Accordingly, for $x \in \{2, 4\}$, if there are two paths p'_1 and q'_1 solving J_x but no two paths p'_2 and q'_2 solving J_{x-1}, instance $I_{\frac{x}{2}}$ can be reduced to $J'_2 := (H_S(\neg s'_1), w, u, t_1, t'_2)$. Finally, if, for $x \in \{2, 4\}$, we find paths solving J_x and paths solving J_{x-1}, the only remaining problem is to find two disjoint paths from the vertices w and u to the vertices t_1 and t'_2 in $H_S(\neg s'_1)$. This problem can be solved in $O(|E_{H_S(\neg s'_1)}|)$ time with the algorithm of Ford and Fulkerson [3].

Case 3 $H_S(\neg s'_1)$ contains neither s'_2 nor t'_2. Then a solution of I_1 exists if and only if there are two paths p and q solving $J_1 = (H_S(s'_1), s'_1, s'_2, u, t'_2)$ or $J_2 = (H_S(s'_1), s'_1, s'_2, w, t'_2)$, since the path of p and q ending in a vertex of $S \setminus \{t'_2\}$ can be extended by a path from u or v to t_1 in $H_S(\neg s'_1)$. Like in Case 1, if one of p and q visits both vertices u and w, it can be shortend so as to visit only one of these vertices and end in this vertex. Since there is no 2-separator separating s'_1 and a vertex of S the instances J_1 and J_2 can be solved in linear time. Accordingly, a solution of I_2 exists if and only if there are two paths p and q solving $J_3 = (H_S(s'_1), s''_1, s'_2, u, t'_2)$ or $J_4 = (H_S(s'_1), s''_1, s'_2, w, t'_2)$. Since there may be 2-separators containing s'_1 separating s''_1 from t_1 our reduction algorithm should be recursively applied to J_3 and J_4. However, as in Case 2, J_3 and J_4 can be splitted into a linear number of subinstances that apart from the 2-separators determined by our algorithm are disjoint. Hence, a solution of J_3 and J_4 can be found in time linear in the number of edges of $H_S(s'_1)$.

Case 4 Finally, let us consider the case, where there is no 2-separator separating s'_1 and t_1. Then I_1 can be solved in linear time. However, as in Case 2 and Case 3, for solving I_2, we should again split I_2 recursively in a linear number of instances on graphs that except of the nodes of a 2-separator are pairwise disjoint.

Theorem 10. *The 2-VDPP on a planar graph can be solved in linear time without computing a combinatioral embedding of G or the triconnected components of G.*

Proof. The reduction algorithm used in our proof of Lemma 9 is applied twice in order to guarantee the existence of three disjoint paths between s_1 and t_1 (in the first run) as well as between s_2 and t_2 (in the second run). If, for a 2-separator $S := \{u, w\}$ and a vertex v, during the second run of our algorithm we split a graph H into the subgraphs $H_S(v)$ and $H_S(\neg v)$, one of these graphs must contain both s_1 and t_1 since s_1 and t_1 are 3-connected. Note that replacing H by $H_S(v)$ or $H_S(\neg v)$ cannot decrease the connectivity between two vertices of these graphs, except between the vertices of S. Therefore, during the second run of our reduction algorithm, s_1 and t_1 will either remain 3-connected or will appear as a 2-separator separating s_1 and t_1. In the latter case we know that there can be no solution of the 2-VDPP.

References

1. A. Aggarwal, J. Kleinberg, and D. P. Williamson, Node-disjoint paths on the mesh and a new trade-off in VLSI layout, *SIAM J. Comput.* **29** (2000), pp. 1321–1333.
2. D. Eppstein, Z. Galil, G. F. Italiano, and A. Nissenzweig, Sparsification–a technique for speeding up dynamic graph algorithms, *J. ACM* **44** (1997), pp. 669–696.
3. L. R. Ford and D. R. Fulkerson, *Flows in Networks,* Princeton University Press, Princeton, New Jersey, 1962.
4. S. Fortune, J. Hopcroft, and J. Wyllie, The directed subgraph homeomorphism problem, *Theoret. Comput. Sci.* **10** (1980), pp. 111–121.

5. T. Hagerup, A very practical algorithm for the two-paths problem in 3-connected planar graphs, Proc. 33rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2007), Lecture Notes in Computer Science Vol. 4769, Springer-Verlag, Berlin, 2007, pp. 145–150.
6. A. Kanevsky, R. Tamassia, G. Di Battista, and J. Chen, On-line maintenance of the four-connected components of a graph, Proc. 32nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 1991), pp. 793–801.
7. S. Khuller, S. G. Mitchell, and V. V. Vazirani, Processor efficient parallel algorithms for the two disjoint paths problem and for finding a Kuratowski homeomorph, *SIAM J. Comput.* **21** (1992), pp. 486–506.
8. C. L. Lucchesi and M. C. M. T. Giglio, On the irrelevance of edge orientations on the acyclic directed two disjoint paths problem, IC Technical Report DCC-92-03, Universidade Estadual de Campinas, Instituto de Computação, 1992.
9. K. Menger, Zur allgemeinen Kurventheorie. *Fund. Math.* **10** (1927), pp. 96–115.
10. H. Nagamochi and T. Ibaraki, Linear time algorithms for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica* **7** (1992), pp. 583–596.
11. T. Ohtsuki, The two disjoint path problem and wire routing design, Proc. Symposium on Graph Theory and Applications, Lecture Notes in Computer Science, Vol. 108, Springer, Berlin, 1981, pp. 207–216.
12. L. Perković and B. Reed, An improved algorithm for finding tree decompositions of small width, *International Journal of Foundations of Computer Science (IJFCS)* **11** (2000), pp. 365–371.
13. Y. Perl and Y. Shiloach, Finding two disjoint paths between two pairs of vertices in a graph, *J. ACM* **25** (1978), pp. 1–9.
14. N. Robertson and P. D. Seymour, Graph minors. XIII. The disjoint paths problem, *J. Comb. Theory, Ser. B*, **63** (1995), pp. 65–110.
15. P. Scheffler, A practical linear time algorithm for disjoint paths in graphs with bounded tree-width, Report No. 396/1994, TU Berlin, FB Mathematik, 1994.
16. A. Schrijver, Finding k disjoint paths in a directed planar graph, *SIAM J. Comput.* **23** (1994), pp. 780–788.
17. P. D. Seymour, Disjoint paths in graphs, *Discrete Math.* **29** (1980), pp. 293–309.
18. Y. Shiloach, A polynomial solution to the undirected two paths problem, *J. ACM* **27** (1980), pp. 445–456.
19. T. Tholey Solving the 2-disjoint paths problem in nearly linear time. *Theory Comput. Systems* **39** (2006), pp. 51-78.
20. C. Thomassen, 2-linked graphs, *Europ. J. Combinatorics* **1** (1980), pp. 371–378.
21. S. G. Williamson, Depth-first search and Kuratowski subgraphs, *J. ACM* **31** (1984), pp. 681–693.
22. G. Woeginger, A simple solution to the two paths problem in planar graphs, *Inform. Process. Lett.* **36** (1990), pp. 191–192.